CS1112 Fall 2012 Project 2 Part A   due Thursday 2/20 at 11pm

You must work either on your own or with one partner. If you work with a partner you must first register as a group in CMS and then submit your work as a group. *Adhere to the Code of Academic Integrity.* For a group, "you" below refers to "your group." You may discuss background issues and general strategies with others and seek help from the course staff, but the work that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is not OK for you to see or hear another student's code and it is certainly not OK to copy code from another person or from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

Objectives

Completing this project will solidify your understanding of Chapters 2 and 3 in *Insight* (`for`-loops, `while`-loops, nested loops). In problem 2 you will make use of several provided graphics functions.

Ground rule

The use of arrays and the `break` and `continue` commands is *not* allowed. In fact, do not use the `break` and `continue` commands in this course.

1 Fixed-point iteration

A *root* of a function $f(x)$ is a value of x that gives $f(x) = 0$. *Fixed-point iteration* is a numerical method for root finding in non-linear equations. The general idea behind fixed-point iteration is simple: (1) start with some initial guess of the root x, (2) evaluate $g(x)$, where g is some function (described below), (3) set x to be $g(x)$, i.e., the value $g(x)$, and go back to step (2). The hope is that g is chosen in such a way that these repeated evaluations of $g(x)$ cause x to converge gradually to a root of the original function f. Neat, isn't it?

How do we come up with g given f? Newton's method provides a simple definition:

$$g(x) = x - f(x)/f'(x)$$

How extraordinary. Does it actually cause x to converge? Suppose we have the nonlinear function $f(x) = x^2 - 2$. One root of f is the square root of 2. According to Newton, if we guess a value x to be the root, then a good next guess will be $x - (x^2 - 2)/2x$. Let's try this in MATLAB with an initial guess of $x = 2$:

```
>> x=2
x = 2
>> x = x - (x^2 - 2) / (2*x)
x = 1.50000000000000
>> x = x - (x^2 - 2) / (2*x)
x = 1.41666666666667
>> x = x - (x^2 - 2) / (2*x)
x = 1.41421568627451
>> x = x - (x^2 - 2) / (2*x)
x = 1.41421356237469
>> x = x - (x^2 - 2) / (2*x)
x = 1.41421356237310
```

Astonishing! In case you're wondering, `sqrt(2)` in MATLAB returns `1.41421356237310`.

Write a script `cubeRoot` to calculate the cube root of an input number n via Newton's method and fixed point iteration. Assume n is positive. *Math primer:* Finding the cube root of a number n is to find x such that $x^3 = n$. Re-arranging the equality gives $x^3 - n = 0$. In other words, we are looking for the root of non-linear function $f(x) = x^3 - n$.

Some notes on your implementation:

- For each iteration, display the current "guess."
- You must determine when your answer is "good enough" so that the program stops and doesn't loop forever! There are different approaches but no, you cannot calculate the root directly and compare it with the approximated values. *Hint:* Notice that as we approach the "real" answer, the change in the guess becomes very small. For example, in finding the square root of 2 with an initial guess of 2, the changes in the subsequent guesses were .5, $\sim .1$, $\sim .002$, and so forth.

Below is the output from an example run of our `cubeRoot` script. Treat this sample output as a loose guideline for how your output should look like. Depending on your stopping criterion, your program may stop with a different number of iterations given the same n and initial guess.

```
Number to find cube root of:  9
First guess of the #'s cube root:  3
   2.33333333333333
   2.10657596371882
   2.08041558960610
   2.08008387595633
   2.08008382305191
The cube root of 9 is 2.08008!
```

Part B of Project 2 will appear in a separate document. Part B has the same deadline as Part A.